日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２００３年　３月２８日

出 願 番 号
Application Number: 特願２００３－０９２３８１

[ST. 10/C]: [ＪＰ２００３－０９２３８１]

出 願 人
Applicant(s): セイコーエプソン株式会社

２００４年　１月３０日

特許庁長官
Commissioner,
Japan Patent Office 今 井 康 夫

出証番号　出証特２００４－３００４７４９

【書類名】　特許願
【整理番号】　J0098698
【提出日】　平成15年 3月28日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/00
【発明者】
　【住所又は居所】　長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内
　【氏名】　礒村　政一
【特許出願人】
　【識別番号】　000002369
　【氏名又は名称】　セイコーエプソン株式会社
【代理人】
　【識別番号】　100066980
　【弁理士】
　【氏名又は名称】　森　哲也
【選任した代理人】
　【識別番号】　100075579
　【弁理士】
　【氏名又は名称】　内藤　嘉昭
【選任した代理人】
　【識別番号】　100103850
　【弁理士】
　【氏名又は名称】　崔　秀▲てつ▼
【手数料の表示】
　【予納台帳番号】　001638
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0014966

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　ベクトルデータのアドレス参照方法

【特許請求の範囲】

【請求項１】　指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、

指標ベクトルの要素を格納する要素格納用レジスタを複数領域に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴とするベクトルデータのアドレス参照方法。

【請求項２】　前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレスを算出することを特徴とする請求項１記載のベクトルデータのアドレス参照方法。

【請求項３】　前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項２記載のベクトルデータのアドレス参照方法。

【請求項４】　前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴とする請求項１記載のベクトルデータのアドレス参照方法。

【請求項５】　前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項４記載のベクトルデータのアドレス参照方法。

【請求項６】　前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴とする請求項１～５のいずれかに記載のベクトルデータのアドレス参照方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ベクトルデータの読み出しあるいは書き込みにおけるアドレス参照方法に関する。

【０００２】

【従来の技術】

従来、画像処理等において、不規則にメモリ上のデータの読み出しあるいは書き込むことが行われており、そのような処理を効率的に行うべく、種々の方法が提案されている。

例えば、画像処理において、１枚の画像データの中で、特定のブロックのデータをメモリからレジスタに格納する場合、ブロックの一行分の画素データをレジスタに格納した後、ブロックの次行の画素データを読み込むために、メモリ上の所定長離れたアドレスが参照される。

【０００３】

このように、複雑なアドレス参照を行うための方法として、参考文献「スーパーコンピュータ」（オーム社、長島重夫、田中義一著）に記載された技術が知られている。

本文献においては、多次元配列データをメモリからベクトルレジスタに読み込んだり、ベクトルレジスタからメモリに書き込んだりする場合に、間接指標ベクトル参照を用いる技術が開示されている。

【０００４】

間接指標ベクトル参照とは、メモリ上の参照するアドレス順序を格納したリスト（指標ベクトル）を用意しておき、そのリストを順に参照することによって、間接的にメモリ上の所定のアドレスを参照する方式である。

このような間接指標ベクトル参照を用いることにより、メモリ上の複雑なアドレス参照を行うことが可能となる。

【０００５】

【非特許文献１】

長島重夫、田中義一著「スーパーコンピュータ」オーム社、ｐ．３５－４１

【０００６】

【発明が解決しようとする課題】

しかしながら、従来の技術においては、指標ベクトルをベクトルレジスタに格納しておく必要があるため、以下のような問題が生じていた。

第１に、指標ベクトルを用意しておく手順が必要となるため、プログラムのコードサイズや処理サイクル数が増加してしまう。

【０００７】

第２に、用意しておく指標ベクトルが増加すると、指標ベクトルを格納しておくベクトルレジスタも増加することから、本来の演算処理に用いるレジスタリソースが不足し、演算効率の低下を招いてしまう。

本発明の課題は、間接指標ベクトル参照をより効率的に行うことである。

【０００８】

【課題を解決するための手段】

以上の課題を解決するため、本発明は、

指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、指標ベクトルの要素を格納する要素格納用レジスタ（例えば、図４のレジスタファイル４０におけるインデックスベクトルを格納しているレジスタ）を複数領域（例えば、上位および下位の領域）に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴としている。

【０００９】

例えば、発明の実施の形態における命令コード内の“エクステンション”によ

って、分割されたいずれかの領域を選択し、指定された領域のコードによって、指標ベクトル（インデックスベクトル）の要素を読み出すメモリ上のアドレスが指定される。そして、指定されたアドレスのデータを読み出していくことにより、指標ベクトルが生成される。

【００１０】

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレス（例えば、ベースアドレス）に対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコード（例えば、上位３ビットあるいは下位３ビット等）と、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレス（メモリ上の参照アドレス）を算出することを特徴としている。

【００１１】

この方法は、発明の実施の形態におけるインデックス修飾アドレッシングに関連する。

また、前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

【００１２】

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴としている。

【００１３】

この方法は、発明の実施の形態におけるポスト・レジスタ・アップデート・アドレッシングに関連する。

また、前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

【００１４】

また、前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴としている。

本発明によれば、ベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割して、それぞれの領域に指標ベクトルの要素となる所定のコードを格納する。そして、分割した領域のいずれかを選択し、その領域に格納されたコードを用いて、所定の指標ベクトルを取得することとしている。

【００１５】

したがって、１つの指標ベクトルを格納するレジスタの領域に、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

【００１６】

【発明の実施の形態】

以下、図を参照して本発明に係るベクトルプロセッサの実施の形態を説明する。

本発明に係るベクトルプロセッサは、指標ベクトルを格納するベクトルレジスタにおいて、その要素レジスタを分割して使用し、アドレッシング機能を拡張している。

【００１７】

したがって、初めに、このような機能を実現するための基本となる考え方について説明する。なお、本発明は、ベクトルプロセッサにおけるロード命令およびストア命令に関するものであるため、これらを中心に説明する。

ベクトルプロセッサにおいて、ロード命令あるいはストア命令のコードタイプには、アドレッシングモード（アドレス参照の方法）に応じて、以下の３種類が規定されている。

【００１８】

図１は、ロード命令あるいはストア命令のフォーマットを示す図であり、（ａ）は、ベース相対アドレッシングに対応するＬＳ０タイプ、（ｂ）は、ポスト・

オフセット・アップデート・アドレッシングに対応するＬＳ１タイプ、（ｃ）は、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに対応するＬＳ２タイプを示している。

【００１９】

図１において、ＬＳ０タイプはスカラーデータのロード命令およびストア命令に対応しており、ＬＳ１タイプおよびＬＳ２タイプは、スカラーデータおよびベクトルデータのロード命令およびストア命令に対応している。

本発明は、ベクトルデータのロード命令およびストア命令を取り扱うものであり、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに関するものであるため、ＬＳ２タイプについて説明する。

【００２０】

図１（ｃ）において、ＬＳ２タイプのフォーマットには、オペコード（opecode）、デスティネーション（dst）、ベース（base）、リピートアマウント（rptamt）、エクステンション（extension）、インデックス（index）の６つのフィールドが含まれている。

オペコードは、命令の内容を示すフィールドであり、ロード命令あるいはストア命令のいずれかを示すコードが含まれている。

【００２１】

デスティネーションは、ロードあるいはストアの対象となるデータが格納されたレジスタのアドレスを指定するフィールドである。即ち、ロード命令の場合、メモリから読み出したデータを書き込むレジスタのアドレスであり、ストア命令の場合、メモリに書き込むデータを読み出すレジスタのアドレスである。

ベースは、基準となるメモリアドレスが格納されたレジスタのアドレスを指定するフィールドである。

【００２２】

リピートアマウントは、ベクトル命令における命令の繰り返し回数（要素データ数）を示すフィールドであり、命令繰り返し回数が“１”である場合はスカラー命令、“１”以外である場合はベクトル命令となる。

エクステンションは、機能拡張用に用意された５ビットのフィールドであり、

本発明においては、後述のように、エクステンションを利用して、分割した要素レジスタのアドレスを指定する。

【００２３】

インデックスは、ベースに示されたアドレスに対し、どのような修飾を施すか、即ち、ベースに示されたアドレスから参照するアドレス順序を示すフィールドである。本発明においては、後述のように、インデックスを利用して、アドレス修飾の拡張仕様を規定する。

なお、インデックスは、上述の指標ベクトルが格納されたベクトルレジスタを指定しており、インデックスを参照することによって、間接指標ベクトル参照が行われる。

【００２４】

続いて、本発明におけるアドレス修飾の拡張仕様について具体的に説明する。

図２は、インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図２においては、アドレス修飾の拡張仕様を説明するため、C言語に準じた記述スタイルを用いている。

【００２５】

図２において、エクステンションが“０００”である場合、インデックスとして指定されたレジスタの内容（指標ベクトル）を符号付きで解釈し、アドレス修飾を行う。

具体的には、ベースに示されたレジスタの値を常に基準とし、その値にインデックスに示されるアドレスを順次加算することにより、アドレス修飾を行う。

【００２６】

また、エクステンションが“００１”である場合、インデックスとして指定されたレジスタの下位１６ビットを符号付きで解釈し、アドレス修飾を行う。

また、エクステンションが“０１０”である場合、インデックスとして指定されたレジスタの上位１６ビットを符号付きで解釈し、アドレス修飾を行う。

さらに、エクステンションが“０１１”である場合、インデックスとして指定されたレジスタの下位１６ビットおよび上位１６ビットをそれぞれ符号付きで解

釈し、交互にアドレス修飾を行う。具体的には、ベクトル命令における繰り返しの実行順序が偶数である場合、下位１６ビットによってアドレス修飾を行い、奇数である場合、上位１６ビットによってアドレス修飾を行う。

【００２７】

次に、ポスト・レジスタ・アップデート・アドレッシングの拡張仕様について説明する。

図３は、ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図３においては、アドレス修飾の拡張仕様を実現するためのプログラミング例を併せて示している。

【００２８】

図３において、エクステンションが“０００”である場合、インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。

具体的には、ベースに指定されたレジスタの値を所定のレジスタ（図３中のパラメータＴの値を格納するレジスタ）に一旦格納し、そのレジスタの値を更新しながらアドレス修飾を行う。

【００２９】

また、エクステンションが“００１”である場合、インデックスとして指定されたレジスタの下位１６ビットを符号付きで解釈し、アドレスの更新を行う。

また、エクステンションが“０１０”である場合、インデックスとして指定されたレジスタの上位１６ビットを符号付きで解釈し、アドレスの更新を行う。

さらに、エクステンションが“０１１”である場合、インデックスとして指定されたレジスタの下位１６ビットおよび上位１６ビットをそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。

【００３０】

ここで、エクステンションが“０００”～“０１１”までの場合、ベクトル命令における命令の繰り返し回数の間、インデックスとして指定された内容に応じてアドレス修飾が行われ、繰り返し回数が終了した後、ベースに指定されたレジスタの値が更新される。

したがって、複数のロード命令あるいはストア命令を用いて、繰り返しロードあるいはストアを行う場合、ベースに指定されたレジスタが自動的に更新されるので、命令発行の都度、ベースのアドレスを別の命令で更新する必要がなく、直ちに命令の実行を行うことが可能である。

【００３１】

また、図３において、エクステンションが“１００”～“１１１”の場合、エクステンションが“０００”～“０１１”の場合とそれぞれ同様にアドレスの更新を行う。ただし、エクステンションが“１００”～“１１１”の場合、命令の繰り返し回数が終了した後、ベースに指定されたレジスタの値を更新することなく命令の実行を終了する。

【００３２】

次に、本発明に係るベクトルプロセッサの構成を説明する。

図４は、本発明を適用したベクトルプロセッサ１の構成を示す図である。

図４において、ベクトルプロセッサ１は、メモリ１０と、メモリ制御部２０と、命令フェッチ部３０と、レジスタファイル４０と、ロードユニット５０と、ストアユニット６０と、演算ユニット７０とを含んで構成される。

【００３３】

メモリ１０は、ベクトルプロセッサ１に与えられる命令コードおよび演算対象となるデータを記憶している。

メモリ制御部２０は、メモリ１０に対するアクセス、即ち、データの読み出しや書き込みを制御する。例えば、メモリ制御部２０は、ロードユニット５０あるいはストアユニット６０によって指定されたメモリ１０のアドレスからデータを読み出したり、メモリ１０から読み出されたデータをレジスタファイル４０に出力したりする。

【００３４】

命令フェッチ部３０は、メモリ制御部２０を介して、メモリ１０から命令コードをフェッチし、一時的に記憶する。

レジスタファイル４０は、３２個のスカラーレジスタＳＲ０～ＳＲ３１と、８個の要素レジスタからなる８個のベクトルレジスタＶＲ０～ＶＲ７を含んで構成

され、メモリ１０から読み出されたデータおよび演算結果を一時的に記憶する。

【００３５】

ロードユニット５０は、命令フェッチ部３０に記憶された命令コードがロード命令である場合に、メモリ１０から命令コードあるいはデータを読み出す処理を行う。

ストアユニット６０は、命令フェッチ部３０に記憶された命令コードがストア命令である場合に、メモリ１０にデータを書き込む処理を行う。

【００３６】

演算ユニット７０は、命令フェッチ部３０に記憶された命令コードが所定の演算命令である場合に、レジスタファイル４０に記憶された所定データを対象として演算処理を行う。

ここで、ロードユニット５０の構成について、詳細に説明する。

図５は、ロードユニット５０の内部構成を示すブロック図である。

【００３７】

図５において、ロードユニット５０は、命令パイプライン制御部５１と、インデックスレジスタ決定回路５２と、デスティネーションレジスタ決定回路５３と、アドレス演算回路５４と、パイプラインレジスタ（ＰＲ）５５，５６と、レジスタ５７とを含んで構成される。

命令パイプライン制御部５１は、ロードユニット５０全体を制御するものである。

【００３８】

インデックスレジスタ決定回路５２は、命令コードのインデックスフィールドに基づいて、インデックスが格納されたインデックスレジスタを選択する信号（インデックスレジスタ選択信号）を生成する。

なお、インデックス修飾アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を保持し、そのアドレスに対する各要素の相対アドレスを指定することで参照アドレスを生成する。即ち、インデックス修飾アドレッシングの場合、要素ごとに相対アドレスを指定するため、インデックスレジスタとしてベクトルレジスタが指定される。

【００３９】

一方、ポスト・レジスタ・アップデート・アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を参照アドレスすると共に、そのアドレスに対する相対アドレスを指定してベースアドレスが更新される。そして、その更新されたベースアドレスに対する、次の要素の相対アドレスを指定することを繰り返して、参照アドレスを生成する。そのため、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタとベクトルレジスタの双方が指定可能である。

【００４０】

デスティネーションレジスタ決定回路５３は、命令コードのデスティネーションフィールドおよびリピートアマウントフィールドに基づいて、デスティネーションアドレスを格納するためのデスティネーションレジスタを選択する信号（デスティネーションレジスタ選択信号）を生成する。

アドレス演算回路５４は、命令パイプライン制御部５１の指示に基づいて、レジスタファイル４０から入力されるベースアドレスおよびインデックスアドレス（インデックスによって指定されるアドレス）に基づいて、ロード命令の対象となるメモリ１０上のアドレス（ロードアドレス）を算出する。

【００４１】

ＰＲ５５，５６は、命令フェッチ部３０から入力されるデスティネーションフィールドおよびリピートアマウントフィールドのコードを一時的に記憶し、パイプライン処理において１サイクル遅延させて、デスティネーションレジスタ決定回路５３に出力する。

レジスタ５７は、命令フェッチ部３０から入力されるベースフィールドのコードを一時的に記憶する。

【００４２】

ここで、図５におけるアドレス演算回路５４の構成について説明する。

図６は、アドレス演算回路５４の構成例を示す図である。

図６において、アドレス演算回路５４は、Ｉレジスタ５４ａと、Ｔレジスタ５４ｂと、マルチプレクサ（ＭＵＸ）５４ｃ～５４ｅと、加算器５４ｆとを含んで

構成される。

【００４３】

Ｉレジスタ５４ａは、レジスタファイル４０から入力されるインデックスアドレスを一時的に記憶する。

Ｔレジスタ５４ｂは、ＭＵＸ５４ｃを介してレジスタファイル４０から入力されるベースアドレスを一時的に記憶する。

ＭＵＸ５４ｃは、レジスタファイル４０から入力されるベースアドレスと加算器５４ｆの出力であるアップデート・ベースアドレスとのいずれかを切り替えて、Ｔレジスタ５４ｂに出力する。

【００４４】

ＭＵＸ５４ｄは、Ｉレジスタ５４ａから入力されるインデックスアドレスの上位あるいは下位のアドレスを選択して加算器５４ｆに入力する。

ＭＵＸ５４ｅは、Ｔレジスタ５４ｂから入力されるベースアドレスと加算器５４ｆの出力であるアップデート・ベースアドレスとのいずれかを切り替えて、ベクトル命令の各繰り返しにおいて、ロード命令を実行するターゲットアドレスとして出力する。

【００４５】

加算器５４ｆは、Ｔレジスタ５４ｂから入力されるベースアドレスの値と、ＭＵＸ５４ｄから入力されるインデックスアドレス上位あるいは下位のアドレスとを加算し、アップデート・ベースアドレスとして出力する。

なお、図６において、ＭＵＸ５４ｃ～５４ｅは命令パイプライン制御部５１によって制御される。

【００４６】

また、アドレス演算回路５４は、インデックス修飾アドレッシングの場合およびポスト・アップデート・アドレッシングの場合それぞれに対応して、所定の動作を行う。

次に、動作を説明する。

初めに、図４を参照して、ベクトルプロセッサ１全体の動作について説明する。

【００４７】

ベクトルプロセッサ１において処理が行われる場合、メモリ制御部２０を介してメモリ１０から命令フェッチ部３０に命令コードが読み出される。

そして、ロードユニット５０、ストアユニット６０および演算ユニット７０のそれぞれに、命令フェッチ部３０から命令コードが出力される。

命令コードが入力されたロードユニット５０、ストアユニット６０および演算ユニット７０は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【００４８】

ここでは、図６を参照しつつ、命令コードがロード命令である場合について説明する。

命令コードのオペコードがロード命令を示している場合（より詳細にはprefixコードが“０００”である場合）、ロードユニット５０が動作する。

まず、ロードユニット５０は、命令フェッチ部３０から受け取ったベースフィールドのコードを、ベースレジスタ・リード選択信号（データの読み出し時にベースレジスタを選択する信号）としてレジスタファイル４０に出力する。ベースレジスタ・リード選択信号は、レジスタファイル４０内のスカラーレジスタＳＲ０～ＳＲ３１のいずれかをベースレジスタとして選択するための信号である。

【００４９】

そして、ベースレジスタ・リード選択信号を受け取ったレジスタファイル４０から、ベースレジスタ・リード選択信号によって指定されたレジスタに記憶されたベースアドレスの値がロードユニット５０に入力される。

レジスタファイル４０から入力されたベースアドレスの値は、アドレス演算回路５４に入力される。

【００５０】

また、インデックスレジスタ決定回路５２は、命令フェッチ部３０から入力されたインデックスフィールドのコードおよびリピートアマウントフィールドのコードを受け取る。そして、インデックスレジスタ決定回路５２は、命令パイプライン制御部５１からの指示に従って、命令コードがベクトル命令であるかスカラ

一命令であるかを判定し、ベクトル命令である場合、インデックスレジスタ選択信号を、リピートアマウントに示される要素データ数分、レジスタファイル４０に順次出力する。このとき、インデックスレジスタ選択信号によって指定されたレジスタがベクトルレジスタである場合、指定されたベクトルレジスタ内の各要素レジスタを特定するための所定の選択信号が出力される。

【００５１】

インデックスレジスタ選択信号が入力されたレジスタファイル４０からは、インデックスレジスタ選択信号に示されるレジスタから、インデックスアドレスの値が、アドレス演算回路５４に順次入力される。

インデックスアドレスの値を受け取ったアドレス演算回路５４は、ベースアドレスの値と、インデックスアドレスの値とからロードアドレスを算出し、メモリ制御部２０に出力する。なお、アドレス演算回路５４の動作については後述する。

【００５２】

また、命令フェッチ部３０から入力されたデスティネーションフィールドのコードおよびリピートアマウントフィールドのコードは、パイプライン処理におけるタイミングを合わせるべく、ＰＲ５５，５６に一旦記憶された後、デスティネーションレジスタ決定回路５３に入力される。

すると、デスティネーションレジスタ決定回路５３は、デスティネーションレジスタ選択信号を、メモリ１０からロードされたデータと同期させてレジスタファイル４０に出力する。

【００５３】

すると、レジスタファイル４０において、メモリ１０からロードされたデータが、所定のデスティネーションレジスタに順次記憶される。

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、アドレス演算回路５４によって出力されるアップデート・ベースアドレスをベースレジスタに書き込む。

【００５４】

したがって、ベースフィールドのコードをレジスタ５７に保持しておき、レジ

スタ５７によって出力されるコードをベースレジスタ・ライト選択信号（データの書き込み時にベースレジスタを選択する信号）として用い、ベースレジスタ・ライト信号（ベースレジスタに対する書き込み指示信号）が入力されることに対応して、ベースレジスタのデータが更新される。

【００５５】

次に、アドレス演算回路５４の動作について説明する。

まず、インデックス修飾アドレッシングの場合について説明する。

インデックス修飾アドレッシングの場合、サイクル“１”において、ベースアドレスの値が、ＭＵＸ５４ｃを介してＴレジスタ５４ｂに格納される。

一方、インデックスアドレスの値は、Ｉレジスタ５４ａに格納される。

【００５６】

次いで、サイクル“２”において、Ｔレジスタ５４ｂのベースアドレスの値と、Ｉレジスタ５４ａのインデックスアドレスの値とが、加算器５４ｆによって加算され、ＭＵＸ５４ｅを介してターゲットアドレス（ロードアドレス）として出力される。

そして、第３サイクル以後は、ベースアドレスの値がＴレジスタ５４ｂに保持された状態のまま、新たにインデックスアドレスが入力され、加算器５４ｆによって、ベースアドレスとの加算が順次行われる。

【００５７】

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部５１が、ＭＵＸ５２ｄを制御することにより、図２に示すインデックス修飾アドレッシングの拡張仕様が実現される。

即ち、エクステンションが“０００”である場合、Ｉレジスタ５４ａに格納されたデータ（ここでは３２ビット）が、そのまま加算器５４ｆに入力される。

【００５８】

一方、エクステンションが“００１”である場合、Ｉレジスタ５４ａの下位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

また、エクステンションが“０１０”である場合、Ｉレジスタ５４ａの上位１

６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

【００５９】

さらに、エクステンションが“０１１”である場合、１サイクル毎に、Ｉレジスタ５４ａの下位１６ビットと上位１６ビットとを交互に選択し、３２ビットに符号拡張されたデータが加算器５４ｆに入力される。

続いて、ポスト・レジスタ・アップデート・アドレッシングの場合について説明する。

【００６０】

ポスト・レジスタ・アップデート・アドレッシングの場合、サイクル“１”において、ベースアドレスの値が、ＭＵＸ５４ｃを介してＴレジスタ５４ａに格納される。

一方、インデックスアドレスの値は、Ｉレジスタ５４ａに格納される。

次いで、サイクル“２”において、Ｔレジスタ５４ｂのベースアドレスの値が、ＭＵＸ５４ｃを介して、そのままターゲットアドレス（ロードアドレス）として出力される。

【００６１】

また、同時に、Ｔレジスタ５４ｂのベースアドレスの値は、加算器５４ｆにも出力され、Ｉレジスタ５４ａに記憶されているインデックスアドレスの値と加算される。

次いで、加算器５４ｆの加算結果は、ＭＵＸ５４ｃを介して、Ｔレジスタ５４ｂに格納され、Ｔレジスタ５４ｂがベースアドレスとして記憶する値が更新される。

【００６２】

そして、第３サイクル以後は、新たにインデックスアドレスが入力され、サイクル“１”，“２”と同様に、Ｔレジスタ５４ｂの出力をターゲットアドレスとして出力すると共に、Ｔレジスタ５４ｂがベースアドレスとして記憶する値が更新される。

その後、実行されているロード命令の最後の要素データのターゲットアドレス

を出力するのと同一のタイミングで、加算器５４ｆの出力がアップデート・ベースアドレスとしてレジスタファイル４０に出力される。

【００６３】

レジスタファイル４０では、ベースレジスタ・ライト信号によって指示されたタイミングで、同時に入力されるベースレジスタ・ライト選択信号によって指定されたレジスタにアップデート・ベースアドレスを記憶する。

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部５１が、ＭＵＸ５２ｄを制御することにより、図３に示すポスト・レジスタ・アップデート・アドレッシングの拡張仕様が実現される。

【００６４】

即ち、エクステンションが“０００”である場合、Ｉレジスタ５４ａに格納されたデータ（ここでは３２ビット）が、そのまま加算器５４ｆに入力される。

一方、エクステンションが“００１”である場合、Ｉレジスタ５４ａの下位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

【００６５】

また、エクステンションが“０１０”である場合、Ｉレジスタ５４ａの上位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

さらに、エクステンションが“０１１”である場合、１サイクル毎に、Ｉレジスタ５４ａの下位１６ビットと上位１６ビットとを交互に選択し、３２ビットに符号拡張されたデータが加算器５４ｆに入力される。

【００６６】

また、エクステンションが“１００”～“１１１”までである場合、エクステンションが“０００”～“０１１”までの場合とそれぞれ同様の機能拡張を行うが、命令パイプライン制御部５１がベースレジスタ・ライト信号を出力することなく、ベースレジスタの更新を行わないように制御する。

以上のように、本実施の形態に係るベクトルプロセッサ１は、インデックスによって指定されたベクトルレジスタの要素レジスタを複数の領域に分割し、分割

した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

【００６７】

したがって、１つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

また、インデックスベクトルを用意する手順としては、１つのインデックスベクトル（指標ベクトル）の場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。

【００６８】

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタが指定され、そのレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

そのため、インデックスレジスタとしてベクトルレジスタを用いる場合、ベクトルレジスタ全てにデータを格納した後でなければ後段の処理を開始できないのに対し、スカラーレジスタを使用できることから、スカラーレジスタにデータを格納した後、直ちに後段の処理が開始できる。

【００６９】

したがって、インデックスベクトル（指標ベクトル）を用意する手順が軽減され、プログラムのコードサイズや処理サイクル数を減少させることが可能となる。

即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【００７０】

なお、本実施の形態においては、命令コードのエクステンションフィールドを利用してアドレッシングの機能拡張を行うこととして説明したが、命令コードにアドレッシングの機能拡張のためのコードを含める場合の他、所定のレジスタを設け、そのレジスタに、アドレッシングの機能拡張のためのコードをセットする

こととしてもよい。

【００７１】

また、本実施の形態においては、アドレッシングの機能拡張のために、レジスタを上位および下位の２つの領域に分割して使用することとして説明したが、３つ以上の領域に分けて使用することも可能である。

さらに、本実施の形態においては、命令コードに含まれるエクステンションによって、その命令全体における要素レジスタのアドレスを固定的に指定することとして説明したが、命令実行中に、要素レジスタの指定パターンを動的に変更することも可能である。

【図面の簡単な説明】

【図１】　ロード命令あるいはストア命令のフォーマットを示す図である。

【図２】　インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図３】　ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図４】　本発明を適用したベクトルプロセッサ１の構成を示す図である。

【図５】　ロードユニット５０の内部構成を示すブロック図である。

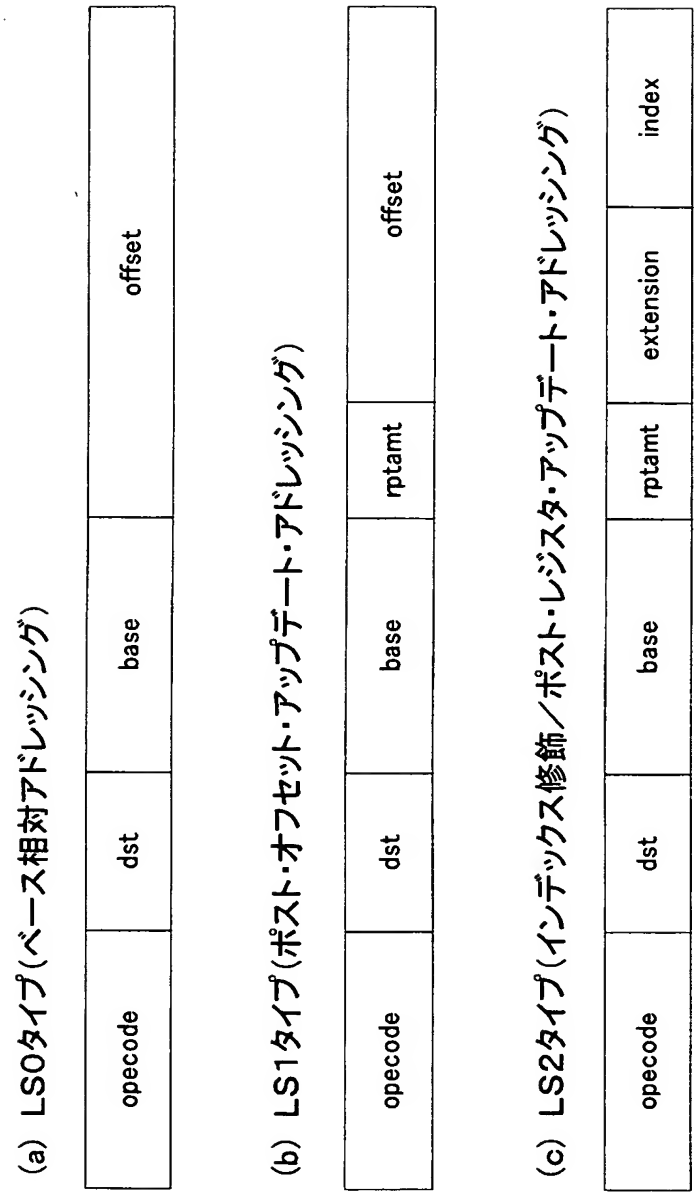
【図６】　アドレス演算回路５４の構成例を示す図である。

【符号の説明】

１　ベクトルプロセッサ、１０　メモリ，２０　メモリ制御部，３０　命令フェッチ部，４０　レジスタファイル，５０　ロードユニット，５１　命令パイプライン制御部，５２　インデックスレジスタ決定回路，５３　デスティネーションレジスタ決定回路，５４　アドレス演算回路，５４ａ　Ｉレジスタ，５４ｂ　Ｔレジスタ，５４ｃ～５４ｅ　ＭＵＸ（マルチプレクサ），５４ｆ　加算器，５５，５６　ＰＲ（パイプラインレジスタ），５７　レジスタ，６０　ストアユニット，７０　演算ユニット

【書類名】　図面

【図１】


(a) LS0タイプ(ベース相対アドレッシング)
opecode
dst
base
offset
(b) LS1タイプ(ポスト・オフセット・アップデート・アドレッシング)
opecode
dst
base
rptamt
offset
(c) LS2タイプ(インデックス修飾／ポスト・レジスタ・アップデート・アドレッシング)
opecode
dst
base
rptamt
extension
index

【図２】

extension	アドレス修飾の方法

000	インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	targetAdrs = base.uw + index[t].h[LOWER];
	}

001	インデックスとして指定されたレジスタの下位16ビット(index[t].h[LOWER])を符号付きで解釈し、
	アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	targetAdrs = base.uw + index[t].h[LOWER];
	}

010	インデックスとして指定されたレジスタの上位16ビット(index[t].h[UPPER])を符号付きで解釈し、
	アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	targetAdrs = base.uw + index[t].h[UPPER];
	}

011	インデックスとして指定されたレジスタの下位16ビット(index[t].h[LOWER])および上位16ビット(index[t].h[UPPER])をそれぞれ符号付きで解釈し、交互にアドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	if ((t & 1)==0)　targetAdrs = base.uw + index[t].h[LOWER];
	else　　targetAdrs = base.uw + index[t].h[UPPER];
	}

【図３】

extension	アドレス修飾の方法

000	インデックスとして指定されたレジスタの内容を符号付きで解釈し、
	アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].w;
	}
	base.uw = T;

001	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[LOWER];
	}
	base.uw = T;

010	インデックスとして指定されたレジスタの上位16ビットの
	index[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[UPPER];
	}
	base.uw = T;

011	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]を
	それぞれ符号付きで解釈し、交互にアドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　if ((t & 1) == 0)　T += index[t].h[LOWER];
	　else　　T += index[t].h[UPPER];
	}
	base.uw = T;

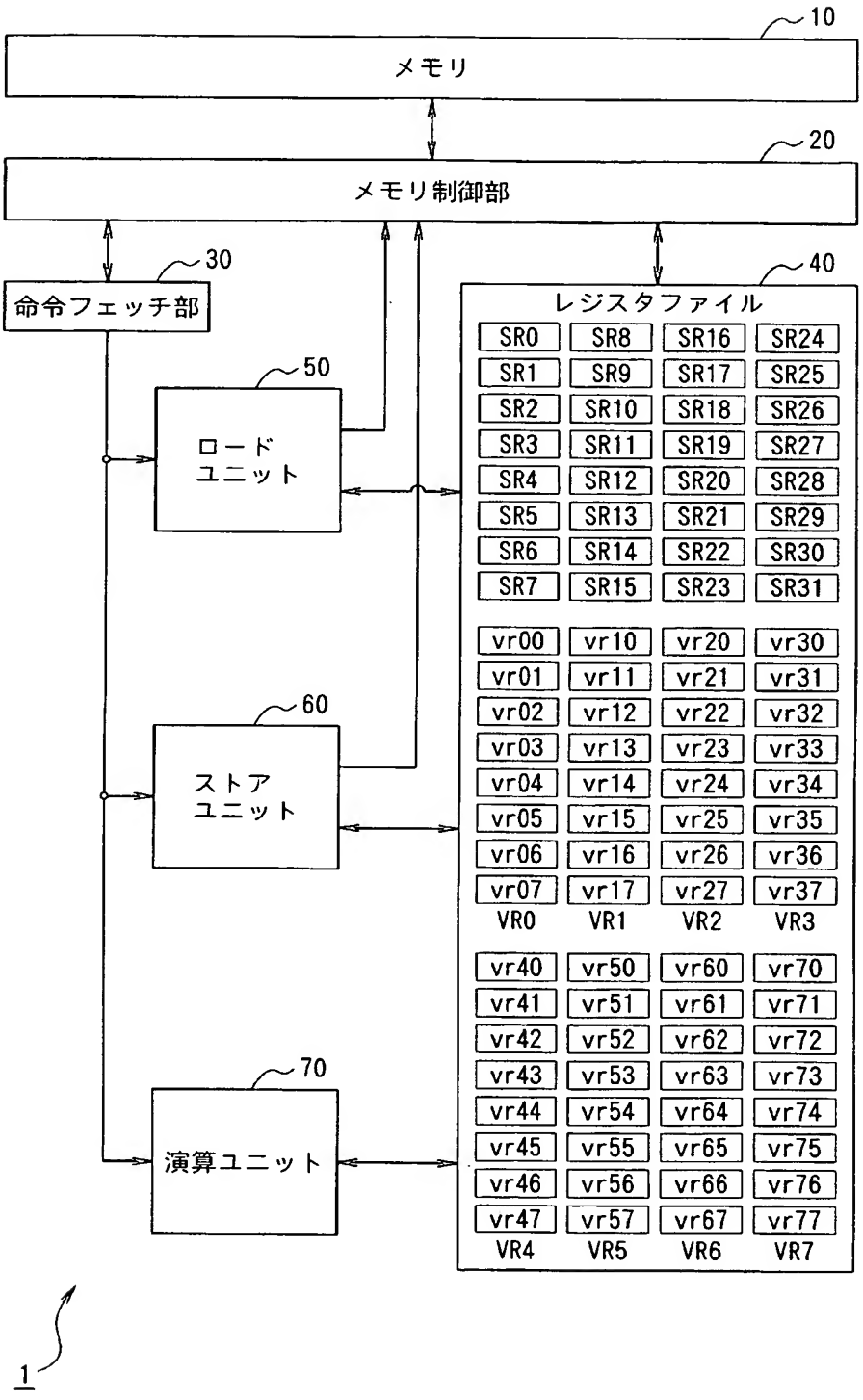
100	インデックスとして指定されたレジスタの内容を符号付きで解釈し、
	アドレスの更新を行う。ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].w;
	}

101	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。
	ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[LOWER];
	}

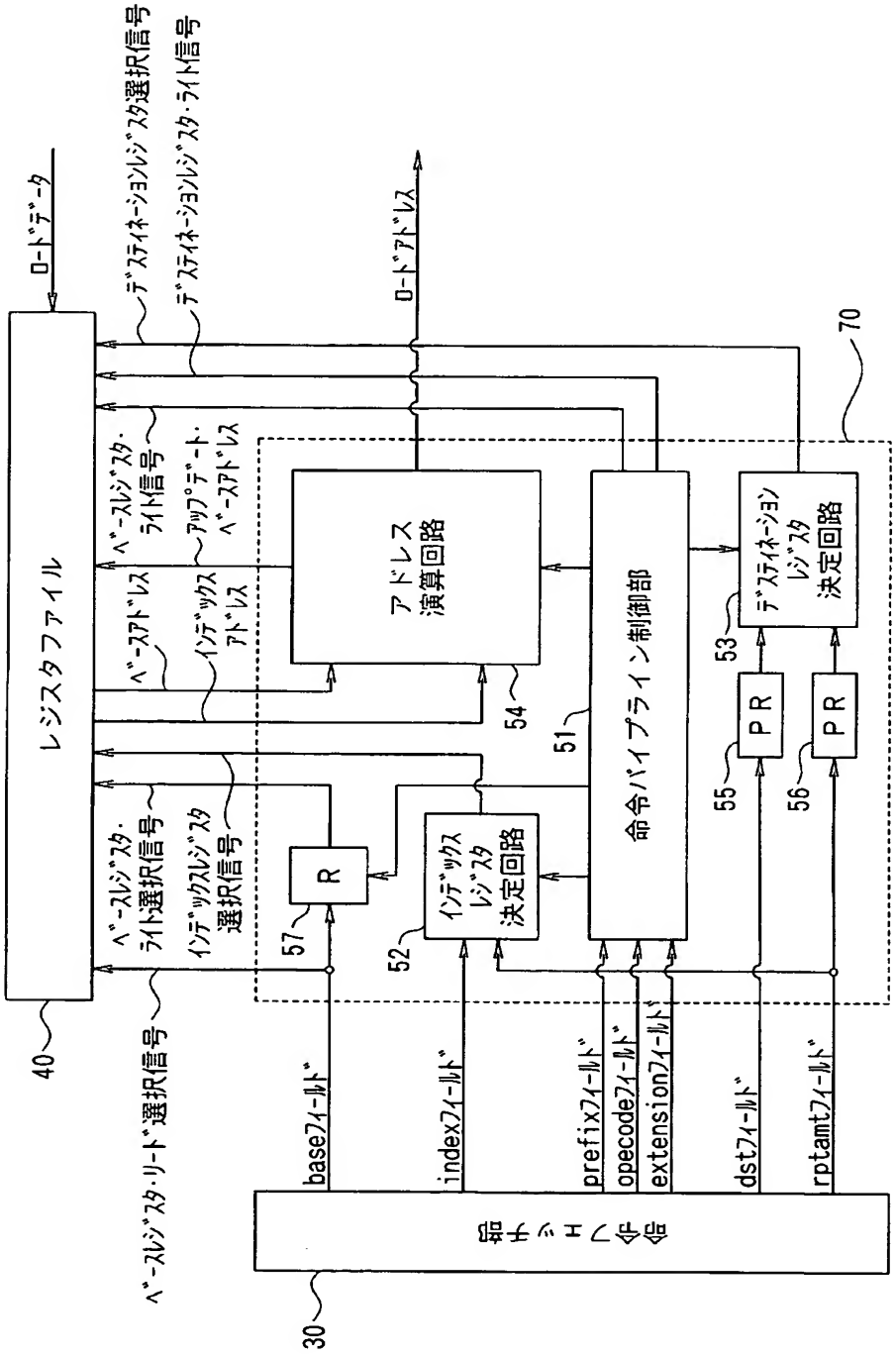
110	インデックスとして指定されたレジスタの上位16ビットの
	index[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。
	ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[UPPER];
	}

111	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]を
	それぞれ符号付きで解釈し、交互にアドレスの更新を行う。
	ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　if ((t & 1) == 0)　T += index[t].h[LOWER];
	　else　　T += index[t].h[UPPER];
	}

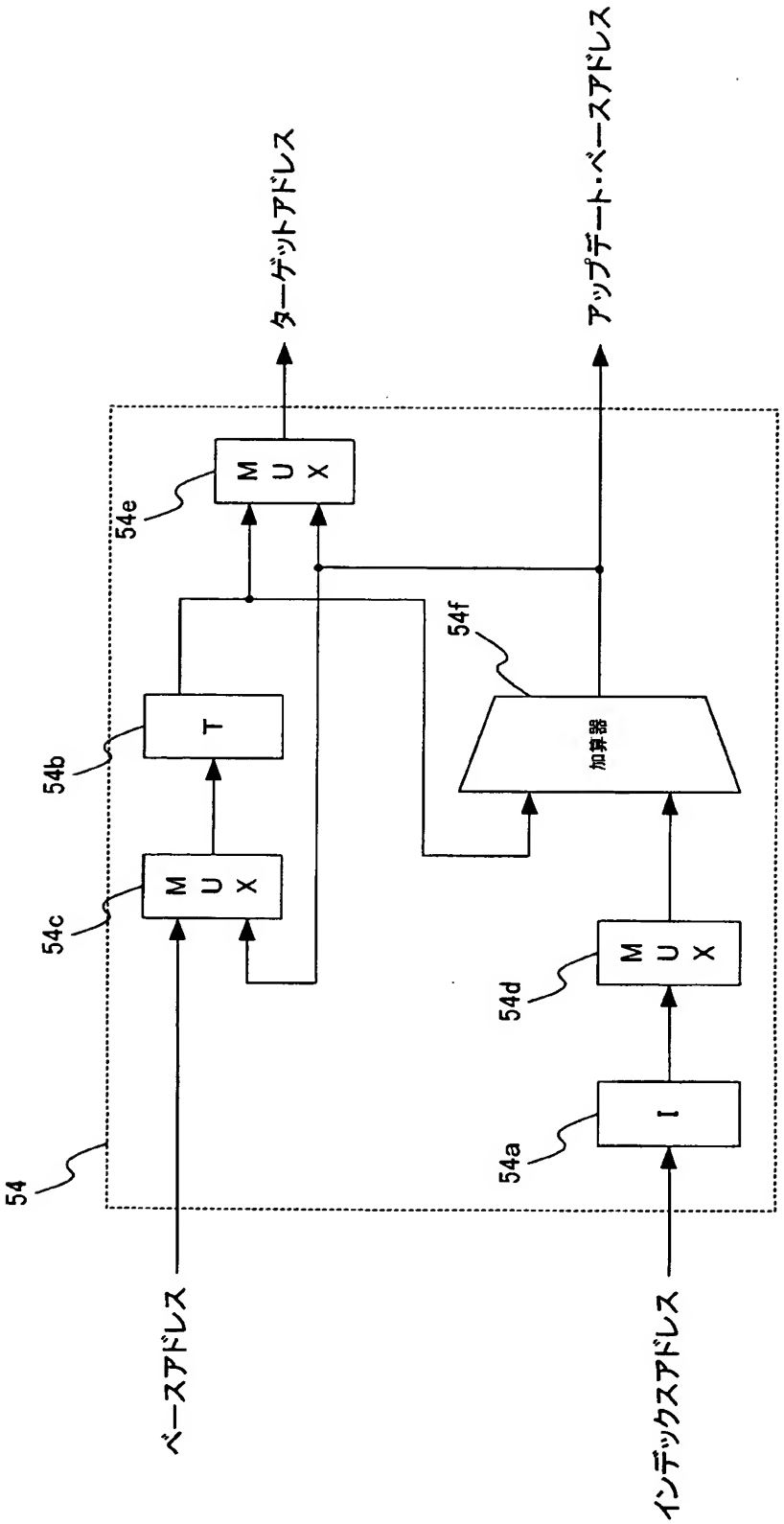
【図４】


10
メモリ
20
メモリ制御部
30
命令フェッチ部
40
レジスタファイル
SR0 SR8 SR16 SR24
SR1 SR9 SR17 SR25
SR2 SR10 SR18 SR26
SR3 SR11 SR19 SR27
SR4 SR12 SR20 SR28
SR5 SR13 SR21 SR29
SR6 SR14 SR22 SR30
SR7 SR15 SR23 SR31
50
ロード
ユニット
vr00 vr10 vr20 vr30
vr01 vr11 vr21 vr31
vr02 vr12 vr22 vr32
vr03 vr13 vr23 vr33
vr04 vr14 vr24 vr34
vr05 vr15 vr25 vr35
vr06 vr16 vr26 vr36
vr07 vr17 vr27 vr37
VR0 VR1 VR2 VR3
60
ストア
ユニット
vr40 vr50 vr60 vr70
vr41 vr51 vr61 vr71
vr42 vr52 vr62 vr72
vr43 vr53 vr63 vr73
vr44 vr54 vr64 vr74
vr45 vr55 vr65 vr75
vr46 vr56 vr66 vr76
vr47 vr57 vr67 vr77
VR4 VR5 VR6 VR7
70
演算ユニット
1

【図５】


ロードデータ
デスティネーションレジスタ選択信号
デスティネーションレジスタ・ライト信号
ロードアドレス
70
レジスタファイル
ベースレジスタ・ライト信号
アップデート・ベースアドレス
ベースアドレス
インデックスアドレス
アドレス演算回路
54
51
命令パイプライン制御部
53
デスティネーションレジスタ決定回路
PR
55
PR
56
ベースレジスタ・ライト選択信号
インデックスレジスタ選択信号
R
57
インデックスレジスタ決定回路
52
ベースレジスタ・リード選択信号
40
baseフィールド
indexフィールド
prefixフィールド
opecodeフィールド
extensionフィールド
dstフィールド
rptamtフィールド
命令デコード部
30

【図６】


ターゲットアドレス
アップデート・ベースアドレス
54e
MUX
54f
加算器
54b
T
54c
MUX
54d
MUX
54a
I
54
ベースアドレス
インデックスアドレス

【書類名】　　要約書

【要約】

【課題】　間接指標ベクトル参照をより効率的に行うこと。

【解決手段】　インデックスによって指定されたベクトルレジスタの要素レジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。したがって、１つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、ベクトルレジスタを効率的に使用することが可能となる。また、インデックスベクトルを用意する手順としては、１つのインデックスベクトルの場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【選択図】　　図２

特願２００３－０９２３８１

出　願　人　履　歴　情　報

識別番号　［０００００２３６９］

１．変更年月日　１９９０年　８月２０日
［変更理由］　新規登録
住　所　東京都新宿区西新宿２丁目４番１号
氏　名　セイコーエプソン株式会社